高级软件实作项目开发文档 题目：基于SpringBoot的食在生鲜超市系统

院系： 软件学院

年级： 19级

专业： 软件工程

成员姓名及学号： 顾思睿20192005273 黄海峰20192005087 王哲孟 20192005343

何昕宇 20192005290

李牧林 20192005323

2022年7月25日

目录

[1.需求分析 3](#_Toc109663458)

[1.1业务需求 3](#_Toc109663459)

[1.1.1背景 3](#_Toc109663460)

[1.1.2 业务机遇 3](#_Toc109663461)

[1.1.3 业务目标 3](#_Toc109663462)

[1.1.4 成功指标 3](#_Toc109663463)

[1.1.5 愿景陈述 3](#_Toc109663464)

[1.1.6 业务风险 4](#_Toc109663465)

[1.1.7 业务假设与依赖 4](#_Toc109663466)

[1.2范围与限制 4](#_Toc109663467)

[1.2.1 主要特性 4](#_Toc109663468)

[1.2.2 初始与后续发布的范围 4](#_Toc109663469)

[1.2.3 限制于排除项 5](#_Toc109663470)

[1.3业务上下文 5](#_Toc109663471)

[1.3.1 干系人资料 5](#_Toc109663472)

[1.3.2项目优先级 5](#_Toc109663473)

[1.4页面设计 6](#_Toc109663474)

[1.4.1主页 6](#_Toc109663475)

[1.4.2个人中心 6](#_Toc109663476)

[1.4.3购物车 6](#_Toc109663477)

[1.4.4订单 6](#_Toc109663478)

[1.4.5付款页 6](#_Toc109663479)

[1.4.6数个具体类别页 7](#_Toc109663480)

[1.4.7待收货 7](#_Toc109663481)

[1.4.8注册登录 7](#_Toc109663482)

[1.5业务规则与用例 7](#_Toc109663483)

[1.5.1业务规则 7](#_Toc109663484)

[1.5.2用例 7](#_Toc109663485)

[2.项目可行性分析 9](#_Toc109663486)

[2.1技术可行性 9](#_Toc109663487)

[2.2经济可行性 9](#_Toc109663488)

[2.3法律可行性 9](#_Toc109663489)

[3.项目设计 10](#_Toc109663490)

[3.1概要设计 10](#_Toc109663491)

[3.1.1组织结构图 10](#_Toc109663492)

[3.1.2功能结构图 11](#_Toc109663493)

[3.1.3 分析类图 11](#_Toc109663494)

[3.2详细设计 12](#_Toc109663495)

[3.2.1类图 12](#_Toc109663496)

[3.2.2活动图 13](#_Toc109663497)

[4.项目实现 18](#_Toc109663498)

[4.1前端 18](#_Toc109663499)

[4.2后端 27](#_Toc109663500)

[4.2.1 SpringCloud 27](#_Toc109663501)

[4.2.2Mysql 37](#_Toc109663502)

[4.2.3Redis 39](#_Toc109663503)

[5.项目分工说明 44](#_Toc109663504)

[5.1组员分工 44](#_Toc109663505)

# 1.需求分析

## 1.1业务需求

### 1.1.1背景

现阶段，国内的疫情形式依旧十分严峻，全国各地仍然存在各种规模的疫情爆发。对于那些正处于疫情防控区内的人，因为疫情而导致无法自由的出行，就意味着无法通过正常渠道来采购生活必需品，例如粮食、蔬果、肉类等，针对这种情况，我们计划开发一个网上生鲜商城，来为这些为了配合疫情防控的居家人员提供正规购买食材的渠道。

### 1.1.2 业务机遇

处于疫情管控区内的人们希望自己的生活质量能够得到保障。网上生鲜商城的出现正好为用户提供了这样一个平台。现在的防疫要求使管控区内的大部分人只能通过政府分配一定量的水和食物来生存，可对于有着特殊情况的家庭来说，是不合理的，该生鲜商城这可以一定程度上为居民提供足够的水和食品，借此保障居民的生活质量。与我们这个生鲜商城合作的商家们也可以通过这种方式来适当弥补疫情带来的经济损失。

### 1.1.3 业务目标

BO-1:在发布前的 6 个月，向各大商家宣传，进行合作。培养配送骑手。

BO-2:在发布后的 6 个月，累计用户数额，并继续扩大合作范围。

BO-3:在发布后的 12 个月，争取向其他疫区城市扩张发展。

### 1.1.4 成功指标

SM-1:在发布后的 6 个月，在本市的疫情管控区取得 50%的蔬菜、生鲜商家的合作。

SM-2:在发布后的 12 个月，能够满足本市疫情管控区内80%居民的基本食材需要。

### 1.1.5 愿景陈述

对于希望想要弥补由于疫情带来的经济损失的商家来说，网上生鲜商城是一个基于互联网的web网页商城，它能够接受居民对于各种生鲜食材的购买，并进行支付处理，然后商家将食材通过骑手配送的方式配送到指定地点，使用网上生鲜商城可以帮助处于疫情管控区内的居民更方便的采购食材。

### 1.1.6 业务风险

RI-1:在与各类符合生鲜商城要求的商家谈判时可能需要大笔资金来进行投入。（概率 =0.8；

影响=6）

RI-2:若营业额较少的商家不同意合作，将降低人们使用生鲜商城的欲望。（概率=0.3； 影

响=3）

RI-3:如果在一段时间后使用生鲜商城的用户较少，会对该软件进行更多的需求开发和使商

家经营重心的改变的投资回报率降低。（概率=0.3； 影响=7）

### 1.1.7 业务假设与依赖

AS-1:生鲜商城为用户提供了恰当的 UI 和交互界面，使用户使用生鲜商城的频率上升。

AS-2:在用户支付后，商家会在约定时间内将商品送达给客户。

DE-1:如果合作的商家自身具有自己的系统可以处理客户的订单，生鲜商城就必须要与之能够形成双向通信。

## 1.2范围与限制

### 1.2.1 主要特性

FE-1:从生鲜商城中选择商品并进行加入购物车和支付

FE-2:创建、查看、修改、删除以及存档各个商家在生鲜商城上显示的食材和展示的内容

FE-3:查看商家上传的食材列表以及营养信息

FE-4:授权顾客能够通过企业内网、智能手机、平板电脑以及外部互联网访问系统

内部

### 1.2.2 初始与后续发布的范围

|  |  |  |  |
| --- | --- | --- | --- |
| 特性 | 发布 1 | 发布 2 | 发布 3 |
| FE-1，购买商品 | 工作时间段配送到指定地点 | 支持各种移动支付，  并且支持信用卡、花  呗贷款支付 | 暂无 |
| FE-2，商品列表 | 新增和编辑查看商品 | 修改、删除和存档商品 |  |
| FE-3，商品成分列表 | 未实现 | 完整实现 |  |
| FE-4，系统访问 | 内网和外部互联网  访问 | 运行于 Windows 的  手机和电脑的应用 | 运行于IOS 以及安卓  的手机和平板电脑  的 APP |

### 1.2.3 限制于排除项

LI-1:生鲜商城的各个商家的商品列表上显示的必须是可以配送的食材，不可外送或者未达到规定时间不外送的食材应该以不可选的方式呈现在商品列表中。

LI-2:该网上生鲜商城在现阶段仅在项目开发的局域网内试行

## 1.3业务上下文

### 1.3.1 干系人资料

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 干系人 | 主要价值 | 态度 | 主要兴趣 | 约束 |
| 商城运营管理  层 | 决定商城未来  的发展前景和  方向 | 强烈承诺支持  发布 2，对于发  布 3 的支持取  决于之前的结  果 | 对项目成本进  行各种估算和  统计 | 无明确约束 |
| 顾客 | 更便捷的生活  方式、节约大  量时间 | 热情高涨，但  可能会因为服  务和价格问题  减少使用 | 方便使用、节  省时间 | 可以在任意设  备上访问该平  台 |
| 商家的服务人  员 | 提高顾客的满  意度，扩大使  用量 | 担心服务不周  被顾客投诉 | 保住工作 | 对服务人员进  行相关的系统  培训 |
| 商家的管理层 | 更高的销售  额，通过曝光  量来增加客户 | 利益至上，一  切均以收益为  首要条件 | 赚钱 | 无明确约束 |

### 1.3.2项目优先级

|  |  |  |  |
| --- | --- | --- | --- |
| 维度 | 约束 | 驱动 | 自由度 |
| 特性 | 所有排入发布 1 的  特性都必须完全可  操作 |  |  |
| 质量 | 用户验收测试的通  过率必须超过 90%，  各项安全测试必须  全部通过 |  |  |
| 成本 |  |  | 在无赞助方评审的  情况下，可以接受  不超过 20%的预算超支 |
| 人员 |  | 项目开发团队包括 1名前端人员，4名后端人员 |  |

## 1.4页面设计

### 1.4.1主页

实现展示外卖数个具体分类别页面，比如美食的日料类别入口、西餐类别入口等等。还有部分热门美食推荐、搜索栏等组件。

### 1.4.2个人中心

实现用户个人中心页面，用户可以在此页面更改个人信息如称呼手机号等等，还可以在此页面查看一些辅助功能如待收货列表。

### 1.4.3购物车

用户选择的餐品放入购物车中暂存，最后用于结算。

### 1.4.4订单

展示用户之前已经选购结算的餐品的具体信息。

### 1.4.5付款页

此页面完成购物车餐品的结算业务。

### 1.4.6数个具体类别页

展示具体的某个餐饮类别（如西餐、日料、韩料）的商店列表，供用户选择。

### 1.4.7待收货

展示用户已付款但未到货的餐品情况。

### 1.4.8注册登录

完成一个用户的登录注册业务。

## 1.5业务规则与用例

### 1.5.1业务规则

BR-1：用户需登录账号才能使用商城功能。

BR-2：用户登录时，如果密码错误，则需重新输入新的验证码。

BR-3：只有具有管理员权限的账号才能进入后台管理。

BR-4：商品库存为零时，用户无法购买该商品。

BR-5：订单超过2小时未支付则取消订单。

### 1.5.2用例

###### 主要操作者和用例

|  |  |
| --- | --- |
| 主要操作者 | 用例 |
| 用户 | 1. 注册 2. 登陆 3. 查看商品 4. 搜索商品 5. 加入购物车 6. 删除购物车 7. 创建订单 8. 删除订单 9. 支付 |
| 商家 | 1. 注册 2. 登录 3. 查询商品 4. 搜索商品 5. 添加商品 6. 修改商品 7. 删除商品 |



# 2.项目可行性分析

## 2.1技术可行性

1. 风险分析：本实作有固定人员5人，不会造成人员流失。本实作成品开发与管理使用springboot、Maven、redis、mysql、javascript、springCloud、Nginx技术，此为成熟的技术，所以并不会有技术性风险。
2. 资源分析：本项目使用eclipse作为IDE进行开发，搭建一个服务性网站，现有的软件和硬件都能支撑项目开发。开发人员已经接受了前后端开发的培训，熟悉开发技术。
3. 技术分析：

1)易用性：基于springboot框架开发的前后端服务性网站，能保证服务器正常运作下99%能成功进入系统。

2)性能：基于express框架开发的网站能承担大约30条并发访问。系统能够存储大约1000条用户信息。

3)防护性：采用post请求模式保证用户的数据在网站不被公开。用户注册登录信息、反馈信息、浏览信息存入本地数据库后保证不被泄露。

4)健壮性：网站设计时，开发人员会考虑各种报错情况并及时给予错误反馈，所以网站遇到非法输入时，能及时做出正确反映，保证系统正常运行而不是崩溃。

## 2.2经济可行性

1. 成本：项目开发中所使用的软件均为免费版，硬件为学生本人计算机，同时使用华为云服务器作为mysql与redis数据库服务器。开发人员为学生故无开发人力成本。开发人员已接受培训，无成本。学生组队进行自我管理，没有管理成本。
2. 效益：此项目为学院实作项目作业，不产生经济效益。
3. 投资回收率：因为没有经济效益，故没有投资回收率。
4. 纯收入：不会产生收入。

## 2.3法律可行性

1. 此实作所使用的数据为网络公开数据，无侵权行为。
2. 此实作的内容积极、向上、健康、非盈利，在法律的运行范围中。

# 3.项目设计

## 3.1概要设计

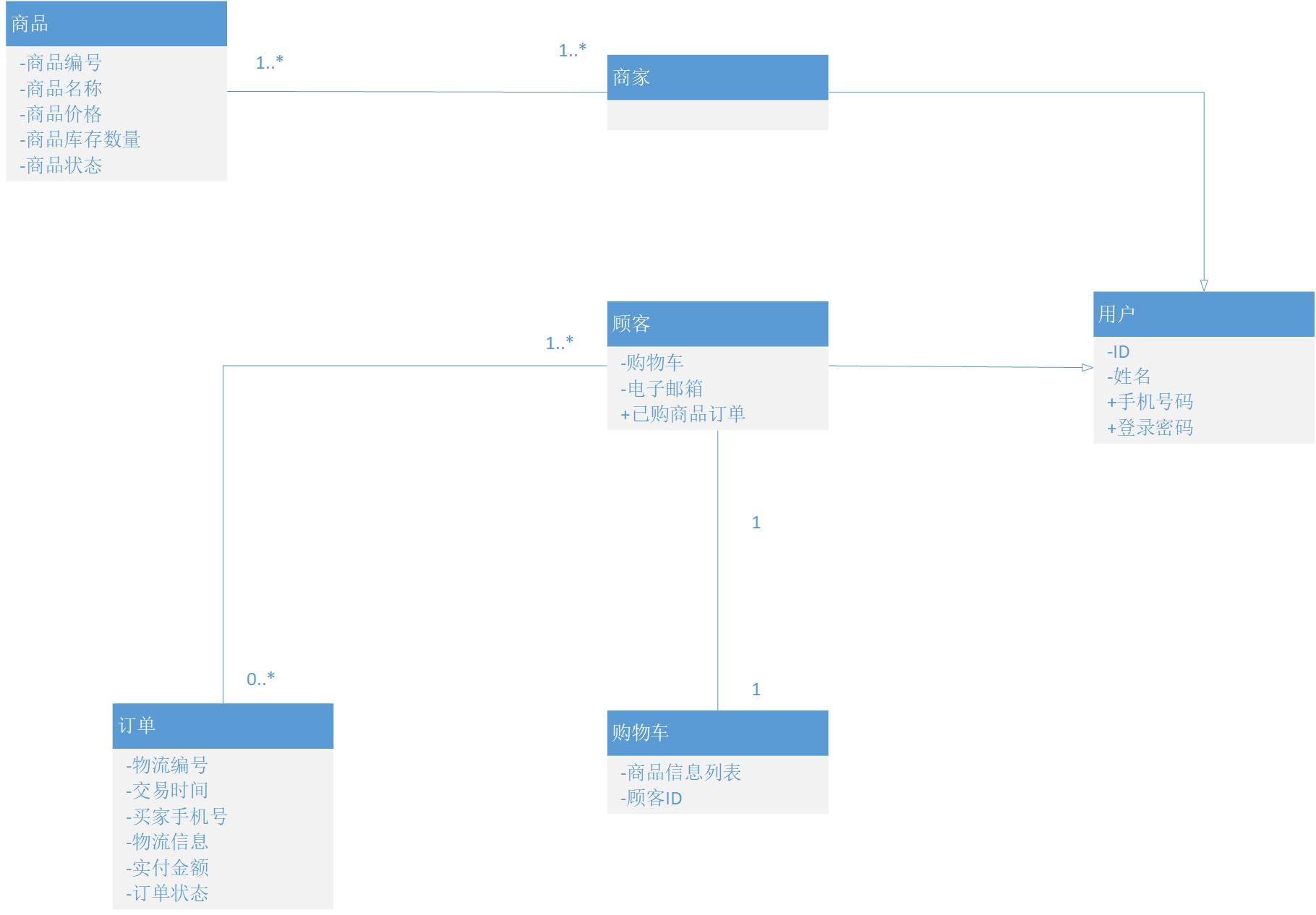
### 3.1.1组织结构图



### 3.1.2功能结构图

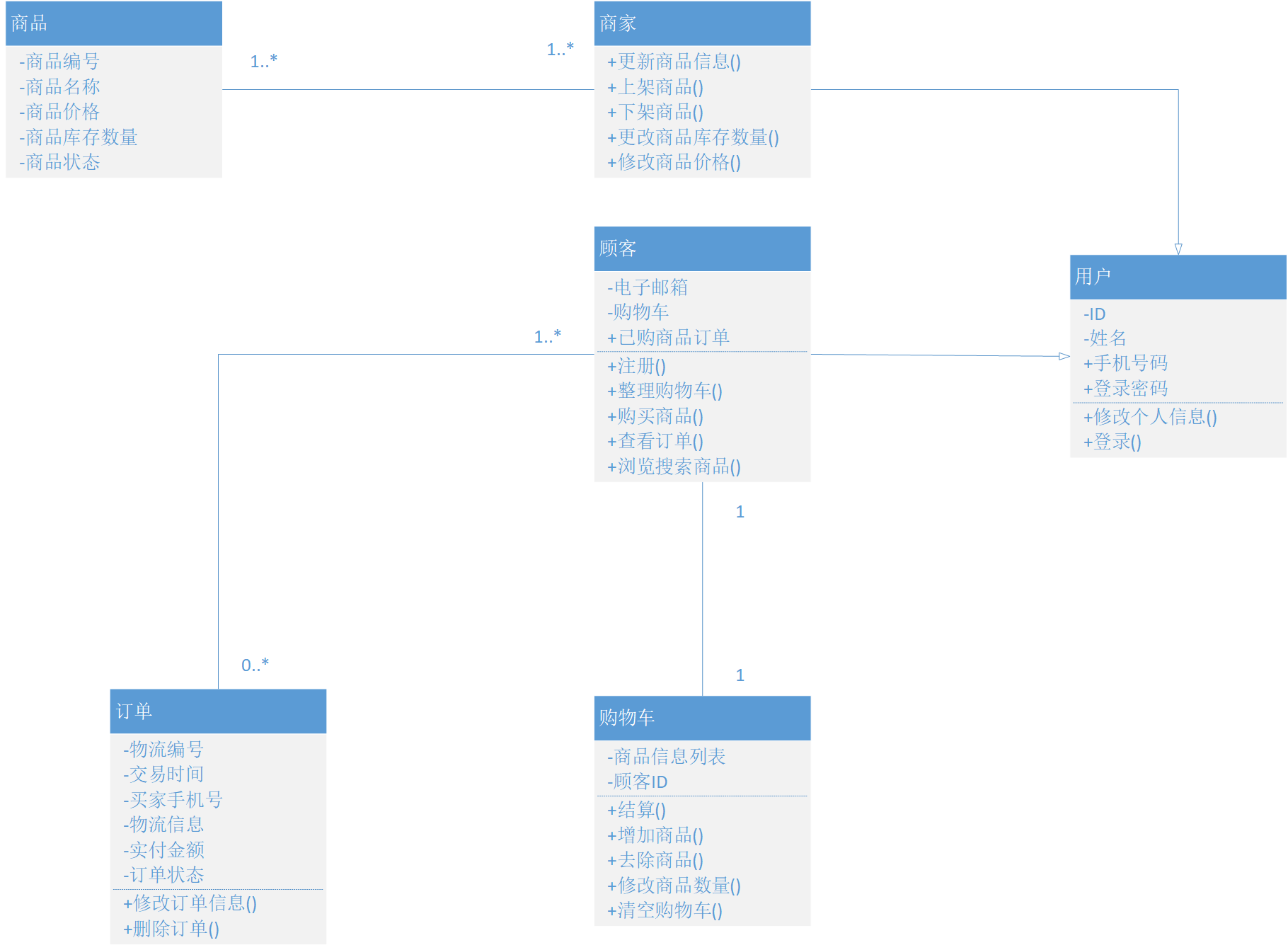


### 3.1.3 分析类图



## 3.2详细设计

### 3.2.1类图



#### 

### 3.2.2活动图

##### 3.2.2.1后台管理活动图



商品新增活动图



商品查询活动图

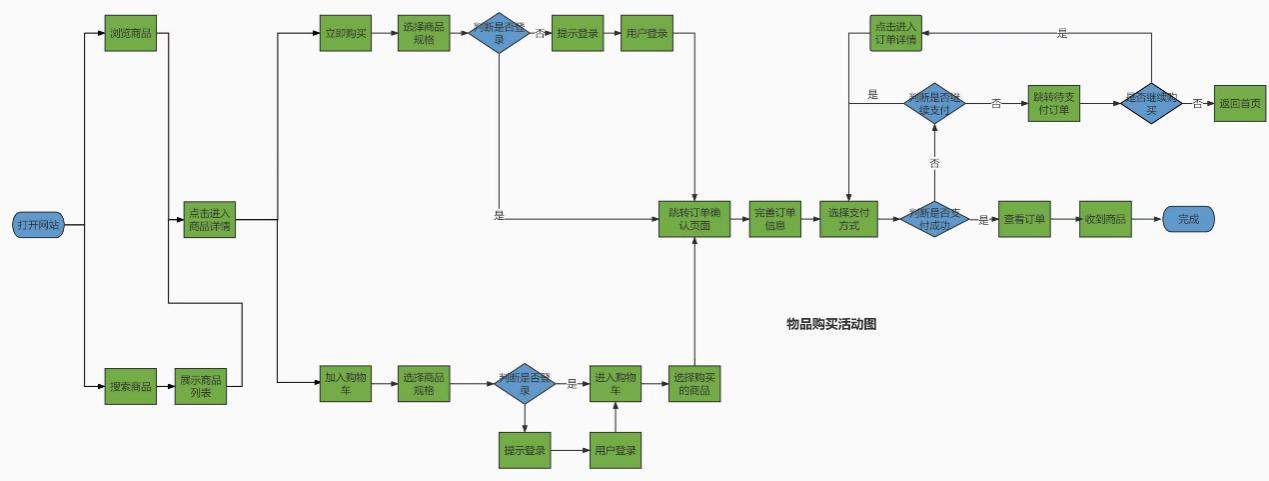


商品信息编辑活动图



商品删除活动图

##### 3.2.2.2物品购买活动图



# 4.项目实现

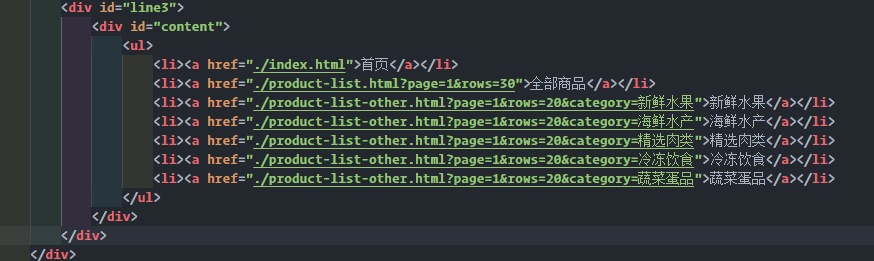
## 4.1前端

该FoodStore生鲜商城商城系统总共分为以下几个模块：

商城index主页、添加商品的manage管理页面、商品页、登录注册页、购物车和订单详情页。

这些模块html页面和各个模块所需要的css文件和jsp文件，共同构成了我们FoodStore生鲜商城的前端页面。接下来就由我来简单分析一下各个模块的相关代码。

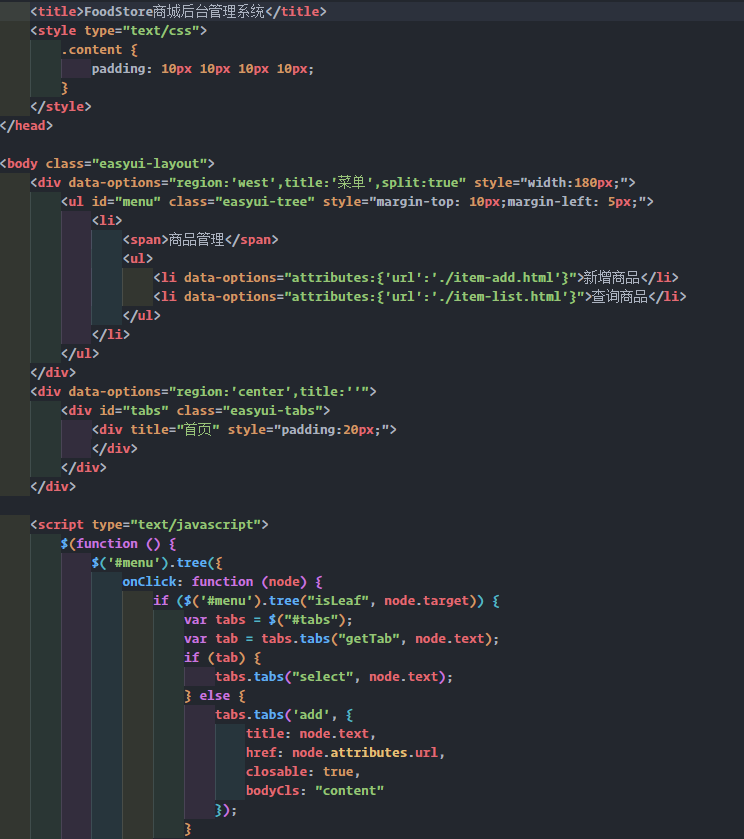
商城index主页：



目的是展示我们商城的大致情况，主要内容是content的目录内容，可以点击并链接到对应的商品分类页中，实现对商品的筛选，方便用户在使用过程中进行商品的购物。



添加商品的manage管理页面：

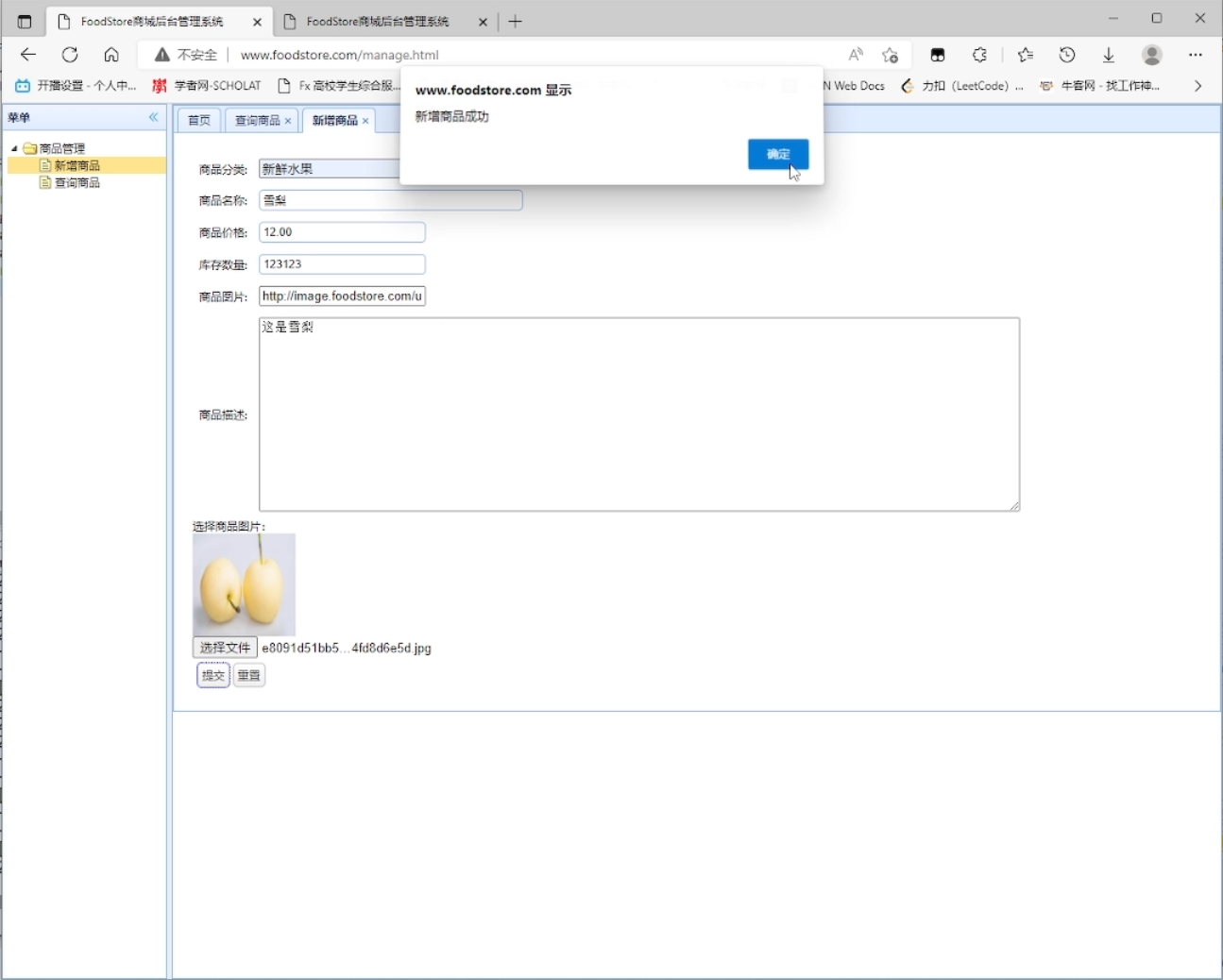


该manage管理页面主要是为了能够对商品进行添加和修改，需要手动输入www.foodstore.com/manage.html进行访问。在该页面中能够跳转到对应的item-add页和item-list页，实现对商品的修改、添加和删除。

在item-add页面中，主要是实现商品添加的功能



新增商品的表单数据作为参数,jquery将其序列化成了key=value&key=value的字符串,在请求体中提交,接收返回的数据,判断执行状态结果弹框提示。

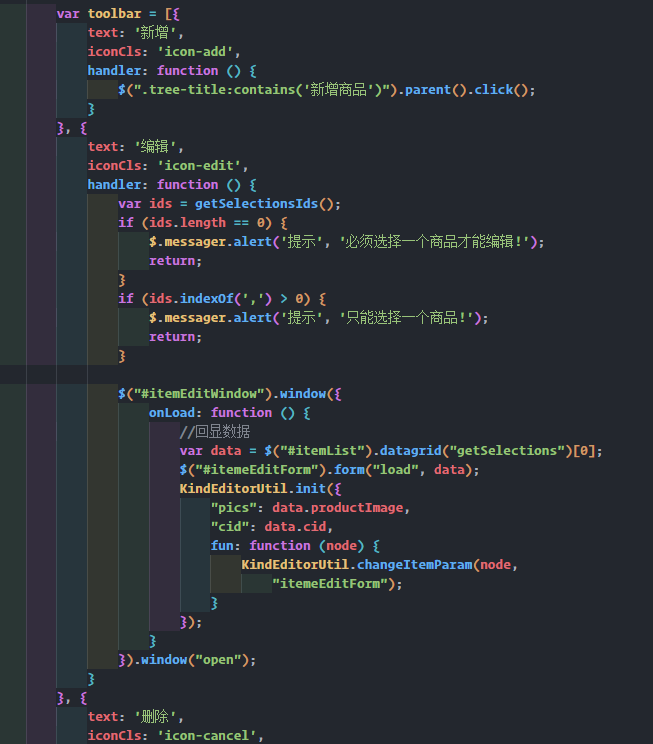


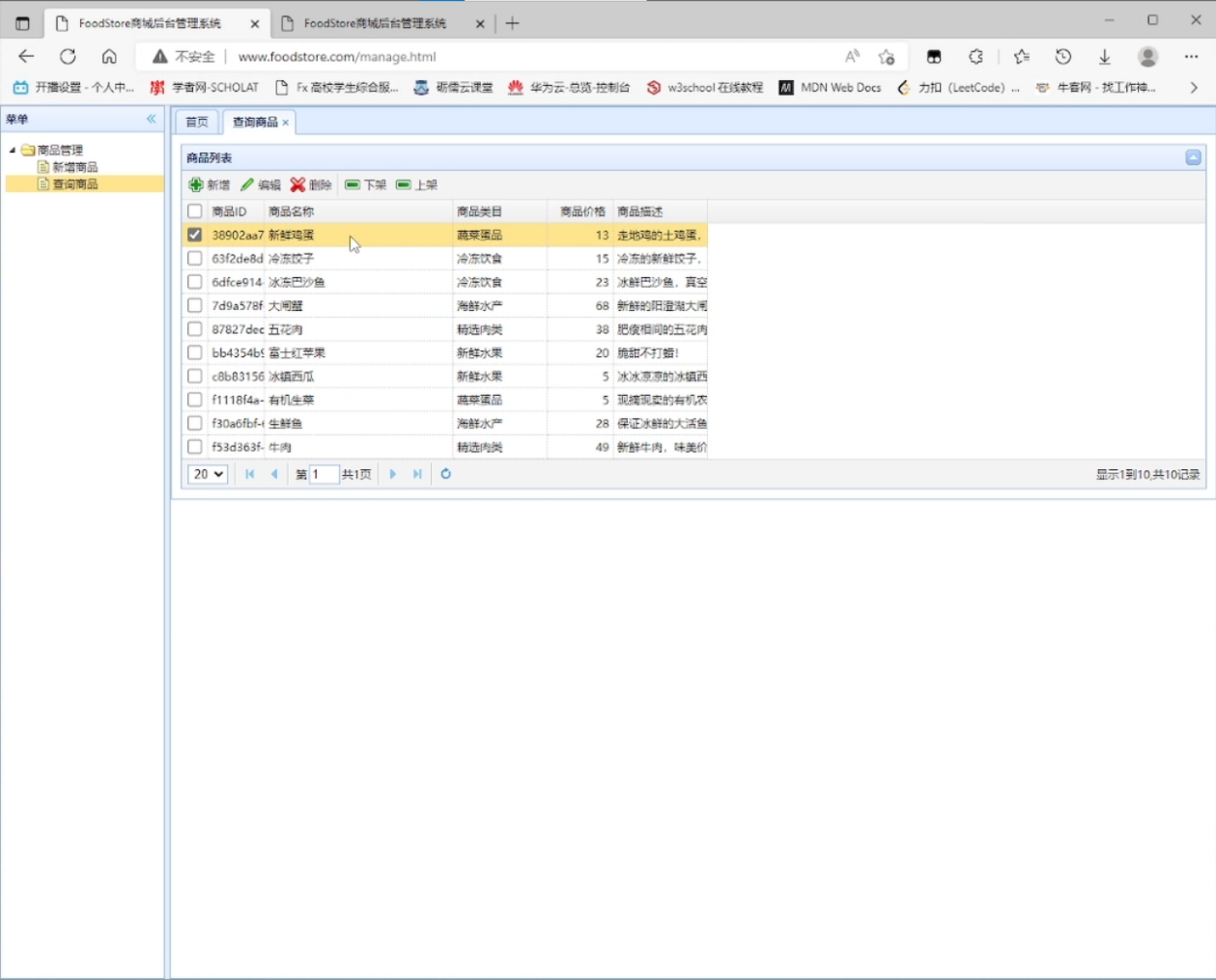
在item-edit页面中，主要是实现修改的功能



这里是将修改商品的表单数据作为参数，后面的逻辑和上述商品添加类似，都是通过js将参数提交给后端接口，再接收后端返回的数据进行判断。

在item-list页面中，主要是实现查询商品的功能，可以看到所有已经在数据库中的商品信息，相关代码如下：





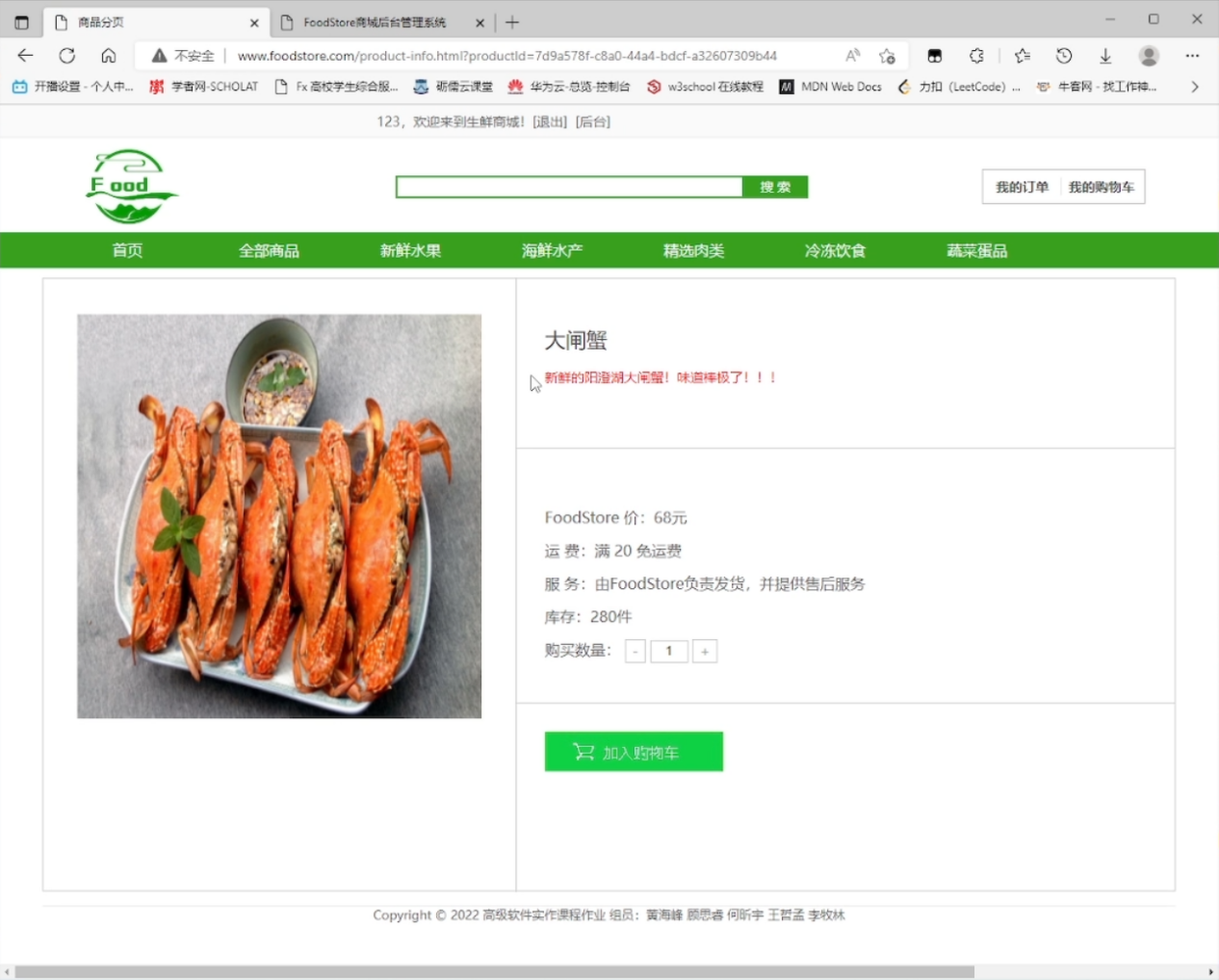
商品页：

商品页分为商品信息页和商品列表页，其中由于后端函数的不同，商品列表页分为2类，一类是只有page和rows两个参数的全部商品列表，另一个是有page、rows和category三个参数的分类商品筛选列表。

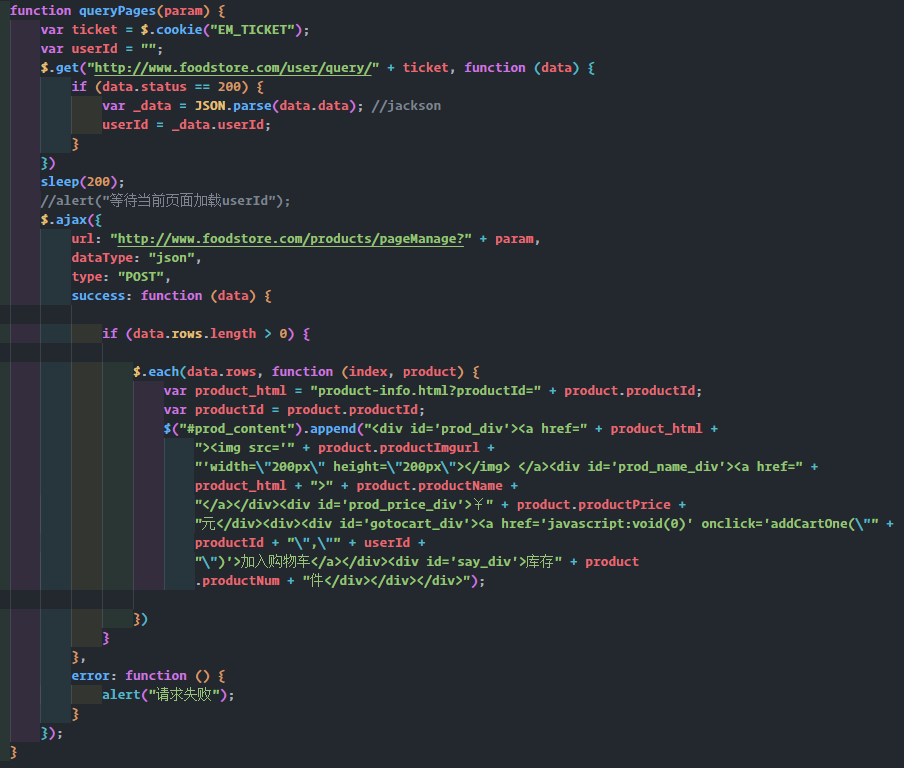
商品信息页：点击查询所选中的单个商品。



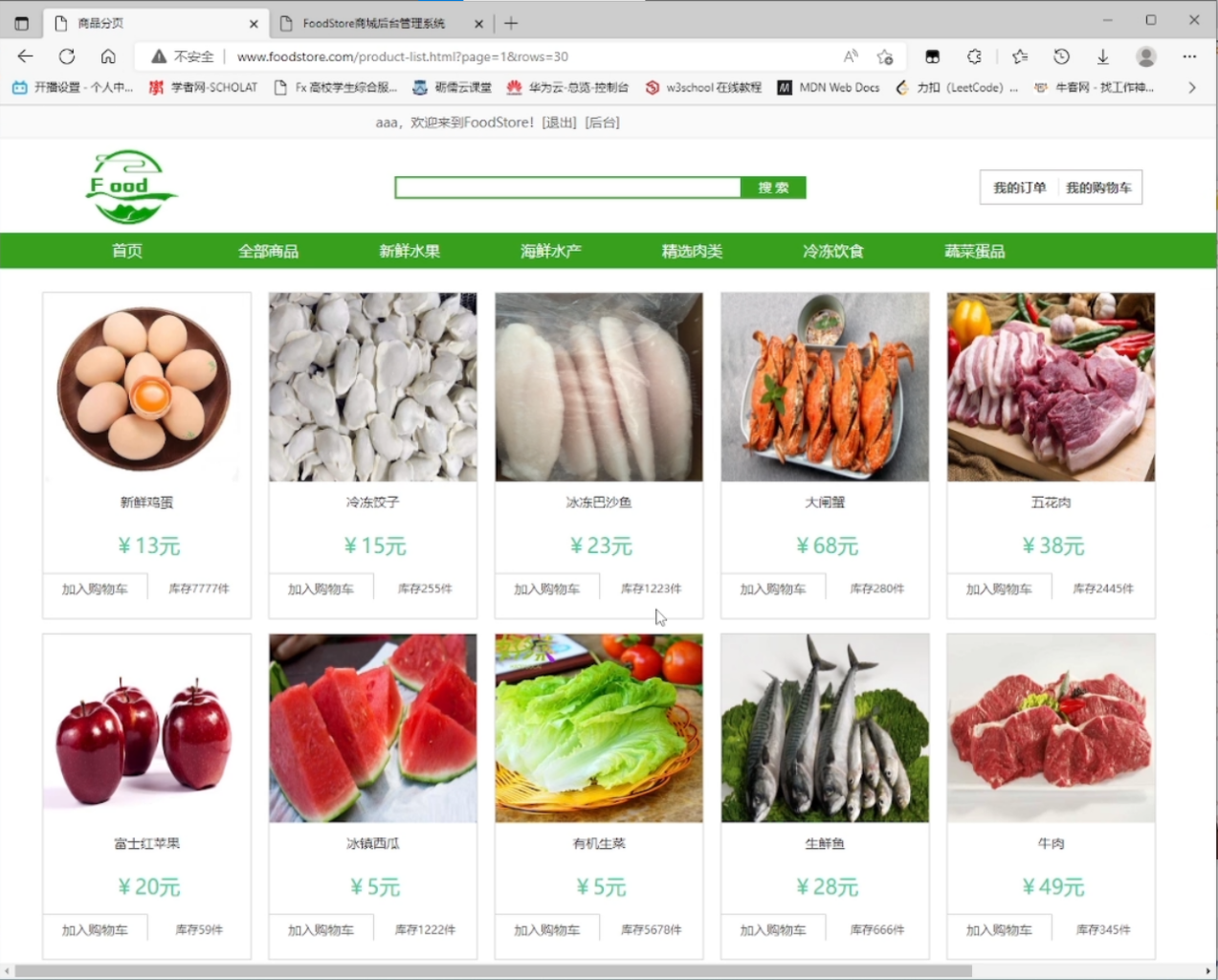
主要逻辑是在页面加载后直接调用ajax请求,将返回的对象数据,解析,调用js代码=拼接到请求地址中后，方便后端接收到productId并返回所查询的product对象json字符串。



商品列表页：将对应的商品展现在网页上。

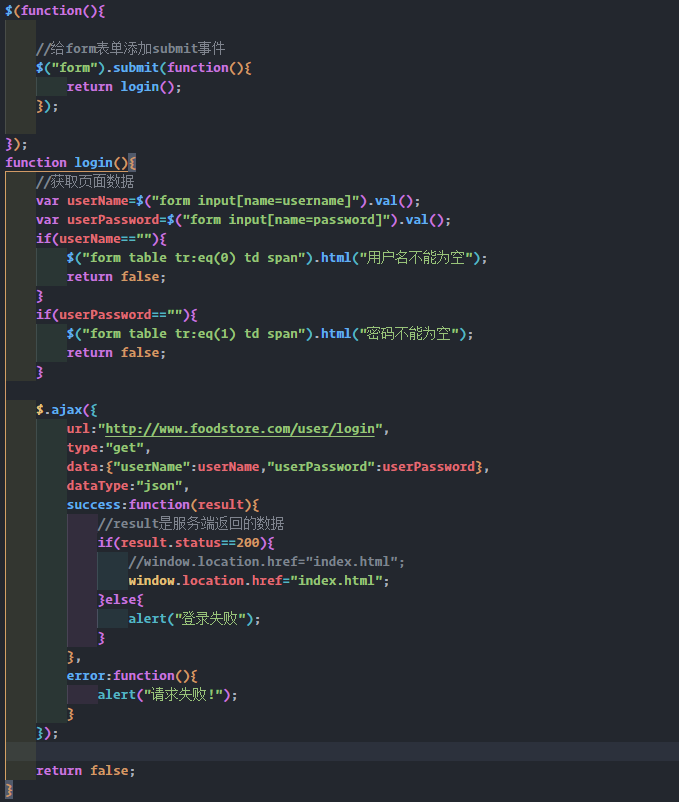


主要逻辑是通过queryPages函数接收数据库中插入的商品数据，如商品的id、img的存储url、name、price等信息。并通过append函数将其动态的生成在prodlist页面，提供给用户进行交互。



登录注册页：让用户能够注册并登录我们的foodstore生鲜商城

Login\_ajax.js



从表单中读取用户名和密码，传到服务端进行验证。通过接收服务端返回的数据来判断是否登录成功，若成功则跳转到index首页，不成功则弹出登录失败。



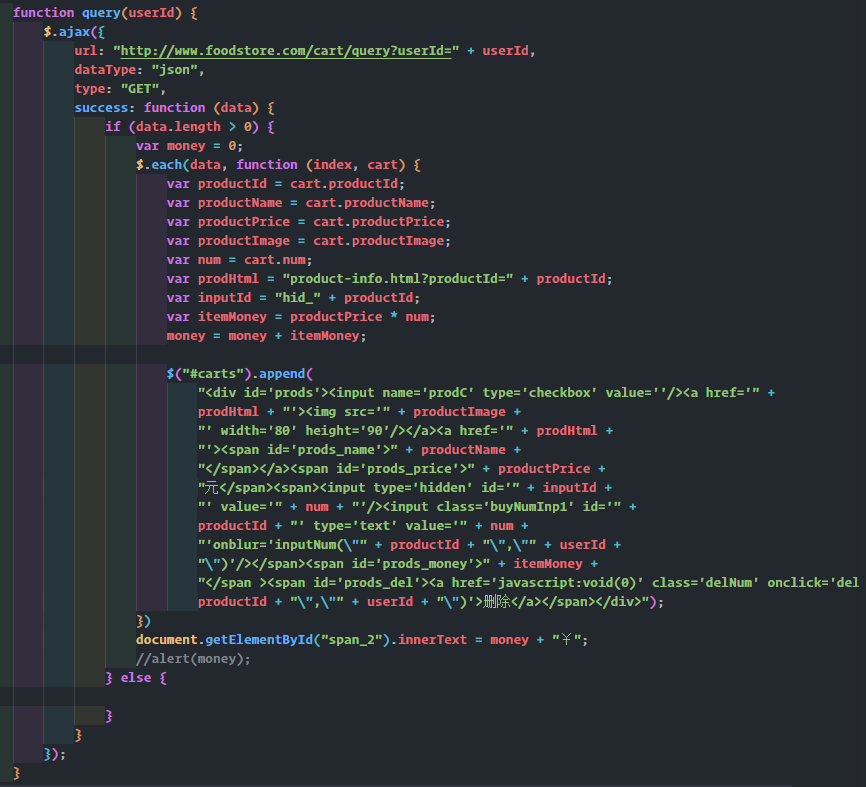
regist\_ajax.js

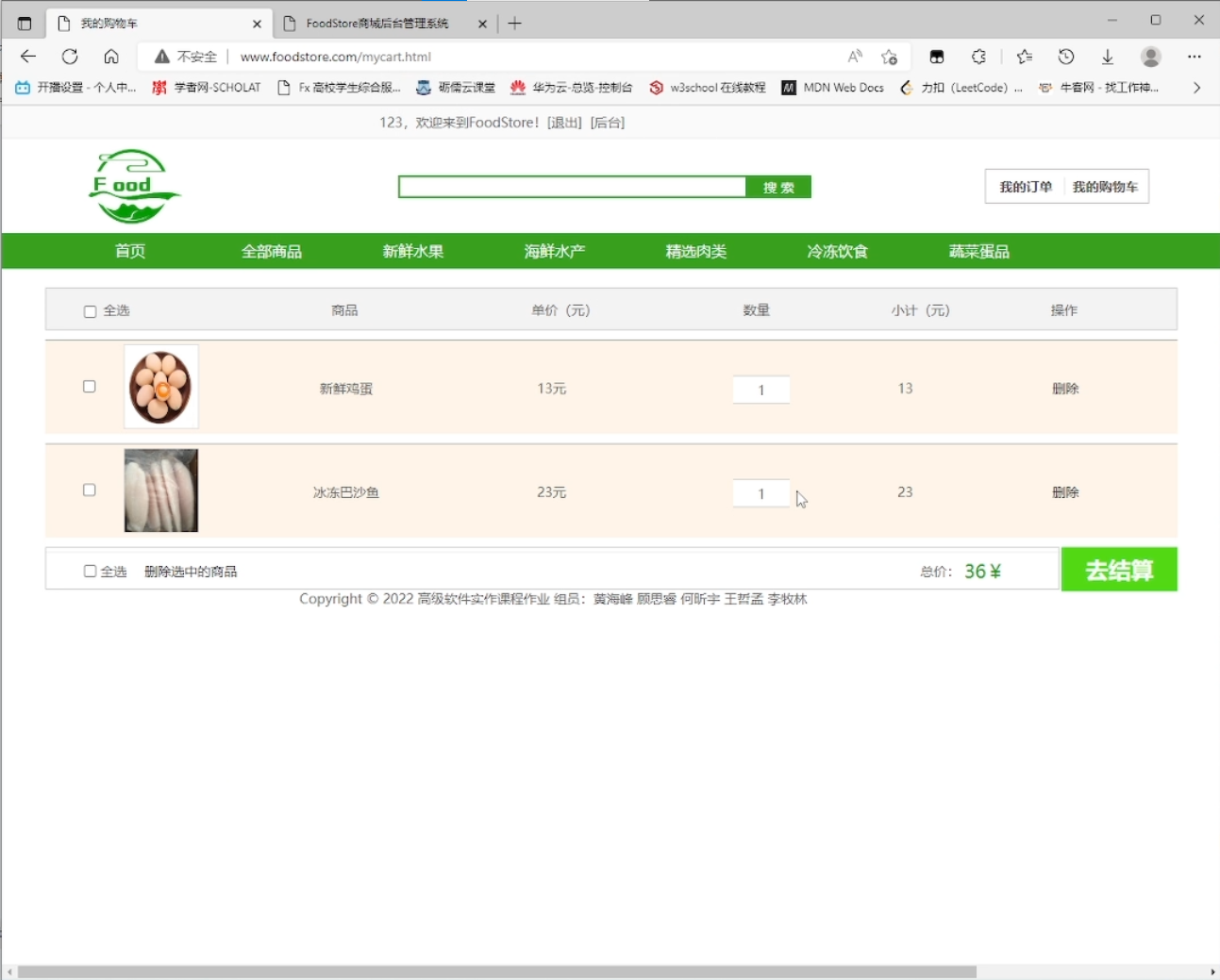


从表单中读取各项输入的数据，例如用户名、用户密码等，然后利用checkForm函数对输入的数据进行判断，若数据格式不符合要求则显示出相对于的警告。如果各项数据格式没有出现问题，就通过url提交到服务端，再通过服务端result的状态判断是否注册成功。

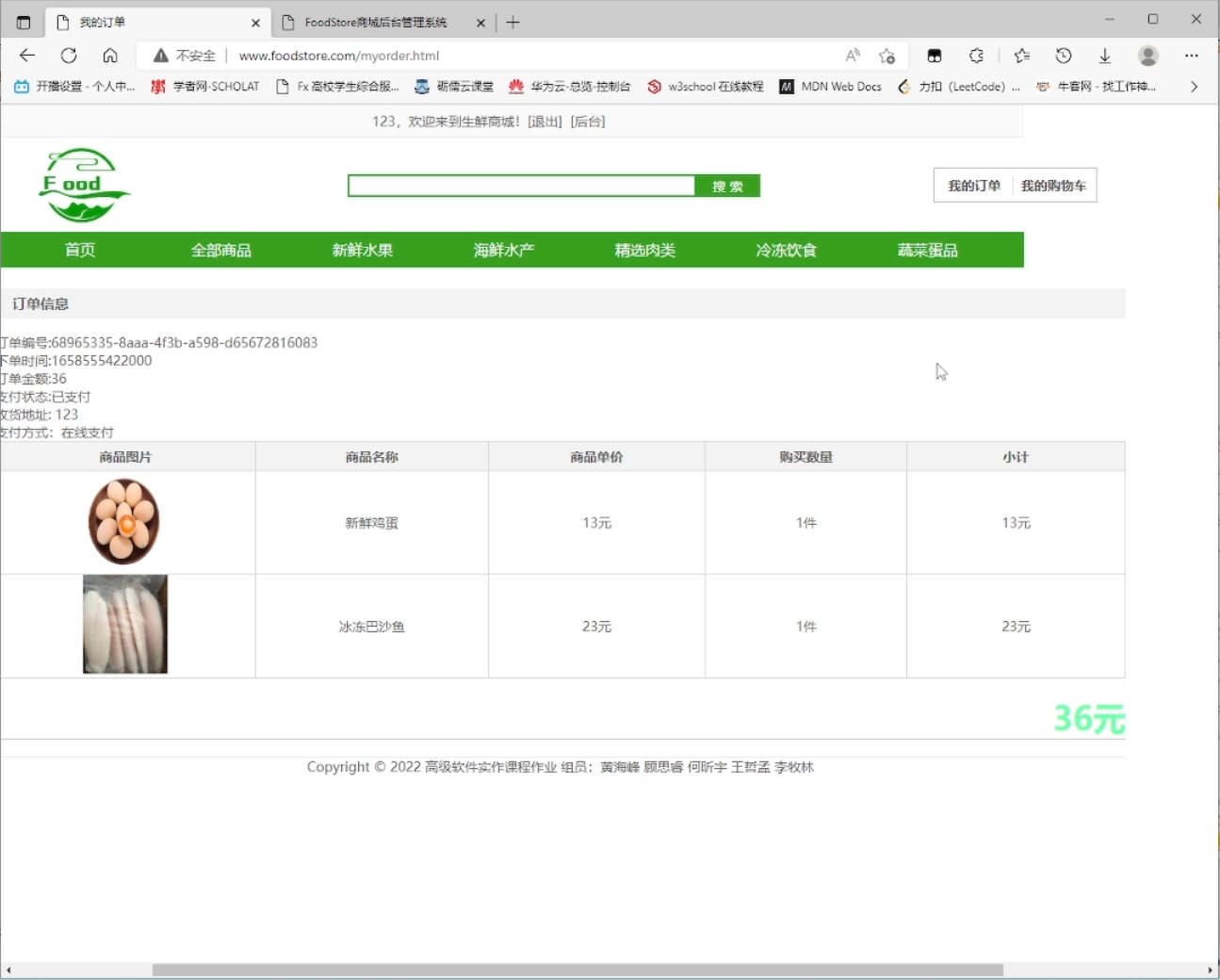


购物车和订单详情页：进行将商品加入购物车并支付完成后生成订单的功能编写



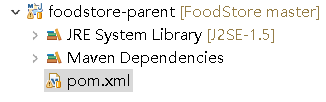
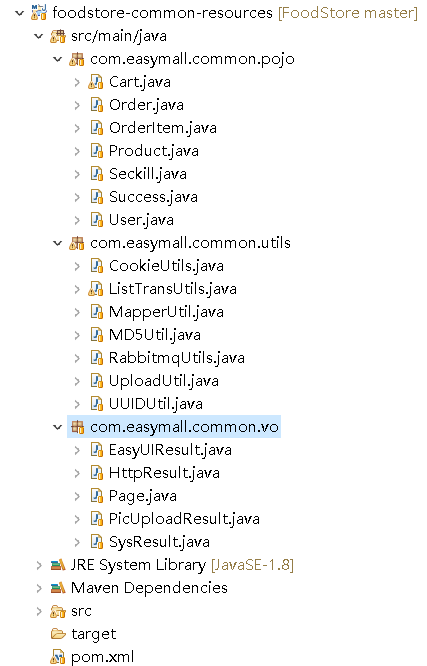
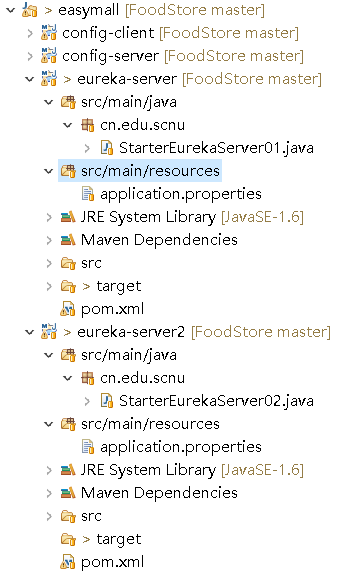
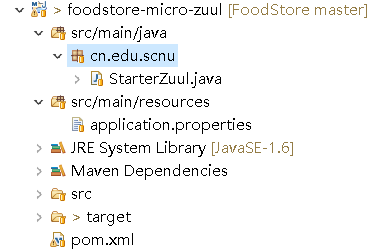
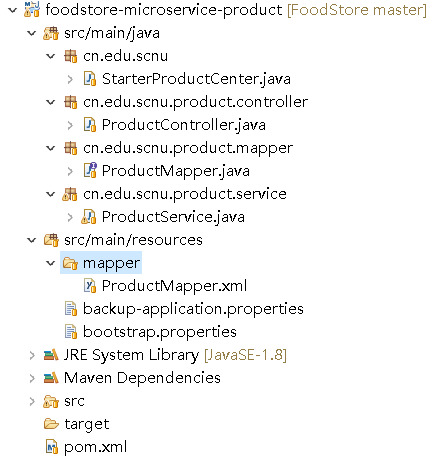
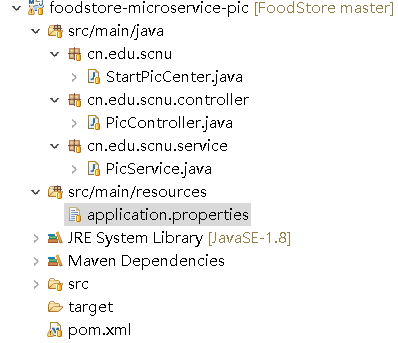
此处是购物车的主要逻辑，意思是根据指定的url地址利用GET方式请求后端数据，请求成功后通过success的data形参来接收数据，再将接收到的数据分别赋值给对应的属性值，最后通过append函数来动态生成购物车的页面  


订单详情页的逻辑和购物车的主要逻辑类似，只不过后端返回的是和订单相关的数据，然后在生成订单详情页面的时候需要书写不同的标签等

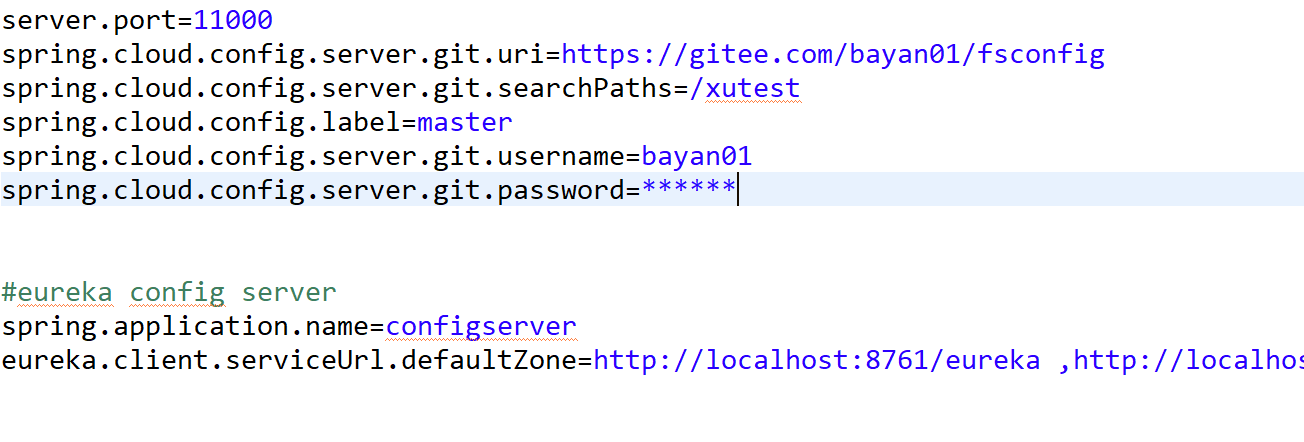


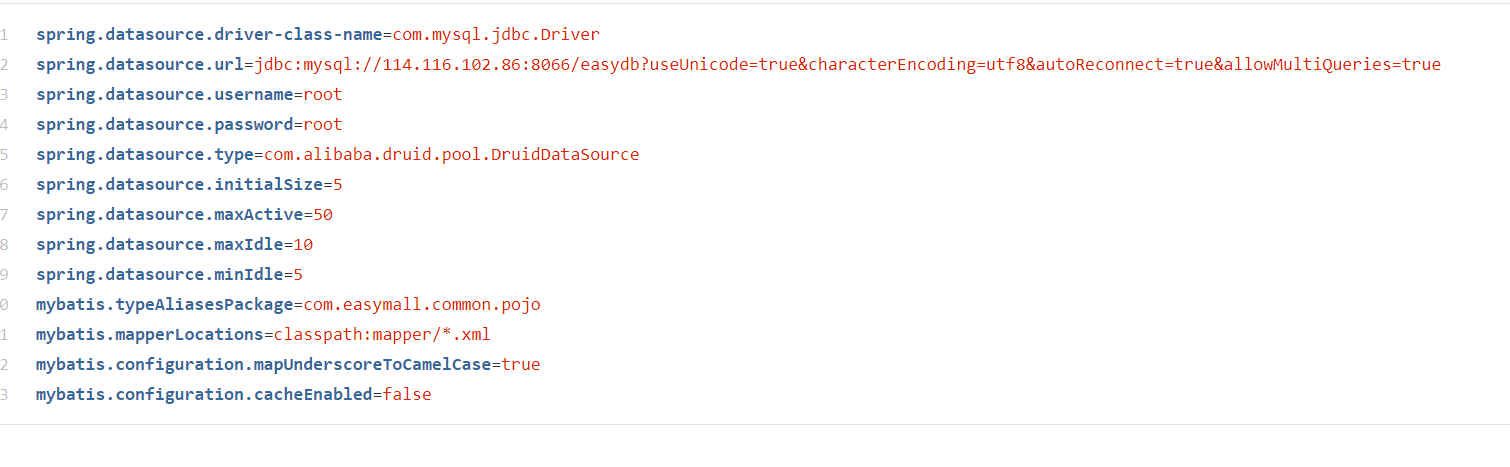
## 4.2后端

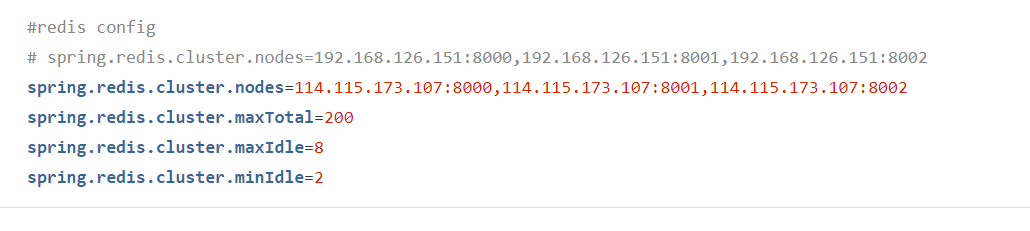
### 4.2.1 SpringCloud

1. SpringCloud框架中依赖资源的管理
   1. foodstore-parent
      1. 
      2. foodstore-parent作为父级工程，合理管理资源。其继承了springboot-parent，导入springcloud资源，并引入了项目中需要用到的公用依赖，包括eureka和ribbon等依赖。
   2. foodstore-common-repository
      1. 
      2. 该工程继承了foodstore-parent，引入了jdbc, mysql, mybatis, druid数据源连接池依赖。创建了自定义初始化属性的配置类。
   3. foodsotre-common-resources
      1. 
      2. 该工程继承了foodstore-parent，维护项目运行所需工具类、视图类与pojo类。
2. 高可用双注册中心的搭建
   1. 
   2. Eureka-server与eureka-server2是相互注册的注册中心，只有端口号不同。以eureka-server为例，其继承了spring-boot-starter-parent，引入了spring-cloud-dependencies管理的依赖，以及spring-cloud-starter-eureka-server依赖，用于启动注册中心服务器。编写了启动类，并在application.properties中进行了属性配置。
3. 高可用zuul网关集群的实现
   1. 
   2. Zuul项目继承spring-boot-starter-parent，引入了spring-cloud-dependencies管理的依赖，并引入了eureka和zuul相关启动依赖。其次，其pom文件中添加了插件简化资源spring-boot-maven-plugin。配置文件中，指定了zuul的端口号，不同微服务的路由，并将标头设置为空。Zuul作为项目与外界的接口，承受连接压力大，需要做集群。把项目打成jar包后，通过命令指定不同端口来运行，即可完成集群搭建。
4. 商品微服务搭建
   1. 
   2. Foodstore-microservice-product微服务继承了foodstore-parent父级工程，将项目管理的三个公共资源全部引入，添加了redis的简化依赖以及spring-cloud-starter-config启动依赖。
   3. 该微服务编写了启动类StarterProductCenter.java，分了控制层、业务层、持久层，完成了商品查询、商品分类查询、后台商品新增、后台商品信息更新、后台商品删除功能。
5. 商品图片微服务搭建
   1. 
   2. Foodsotre-microservice-pic微服务继承了foodstore-parent父级工程并引入了foodstore-common-resources公共资源。
   3. 该微服务编写了启动类StartPicCenter.java，分了控制层与业务层，完成了图片的上传功能。配置文件中指定了端口号，微服务名称，图片存储路径以及图片的url前缀。
6. config配置中心的搭建：

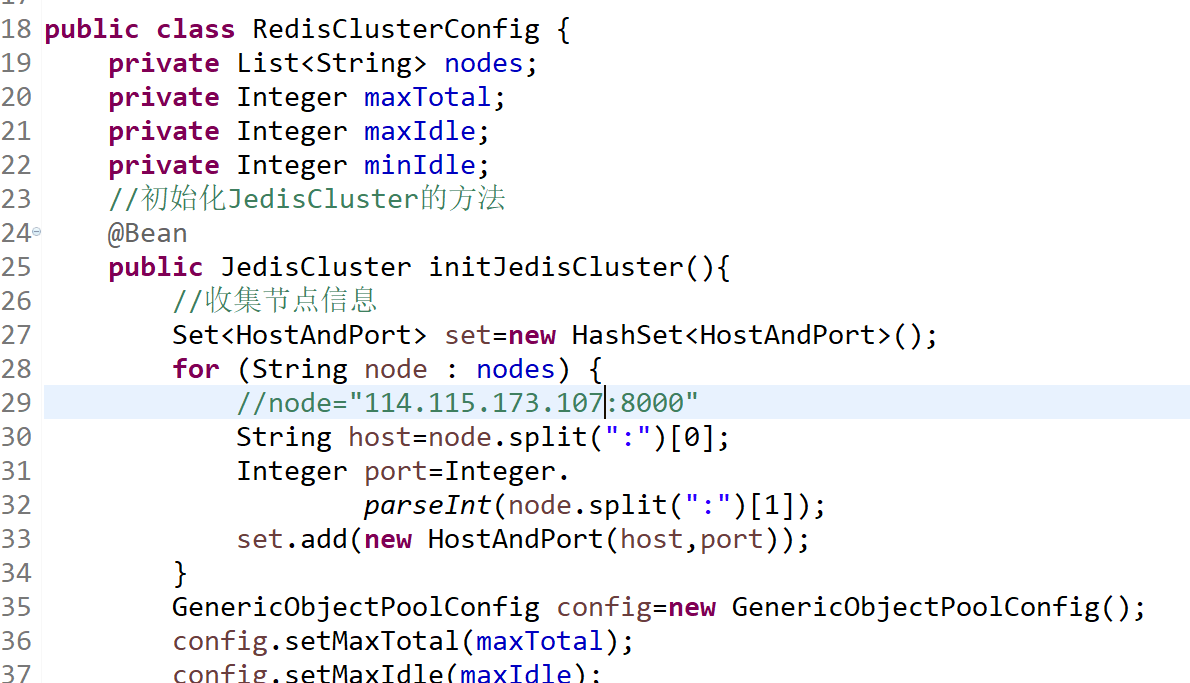
（1）设置配置文件，通过访问git读取配置，将配置中心注册在eureka中心，形成服务提供者，为其他服务提供高可用的访问获取git中properties的配置文件资源。



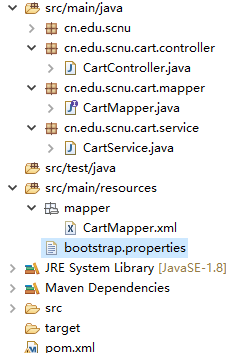




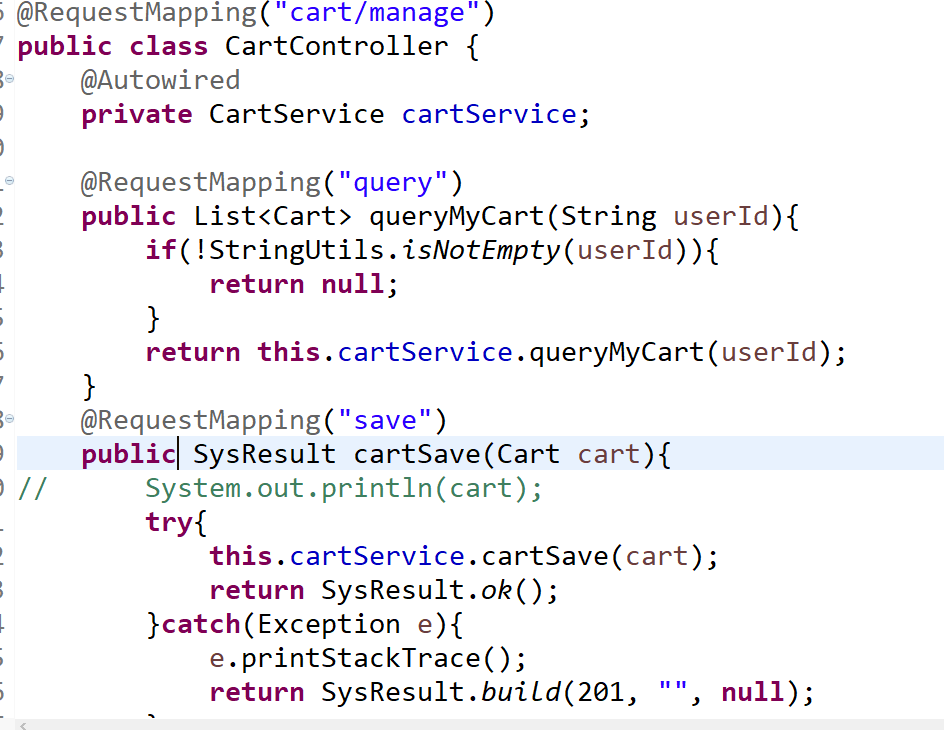
1. 将其他微服务整合到config配置中心作为其客户端。
2. 公用工程rediscluster
3. 将操作rediscluster集群的工作整合到springboot，配置连接信息，并创建配置类。



1. Jedis通过一个节点从redis服务器获取整个集群的信息，这些信息包括节点与其对应连接池的映射关系和槽位与槽位所在节点对应连接池的映射信息，将信息保存在JedisClusterInfoCache中。发送请求时，JedisCluster对象从初始化得到的集群map中获取key对应的节点连接，即一个可用的Jedis对象，然后通过这个对象发送操作命令。
2. 购物车系统的实现
3. 配置各种文件

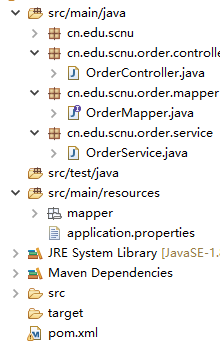


1. 建立dao层，service层，controller层，实现购物车商品展示，购物车商品添加，购物车商品删除，购物车商品数量修改以及商品批量选择的功能。

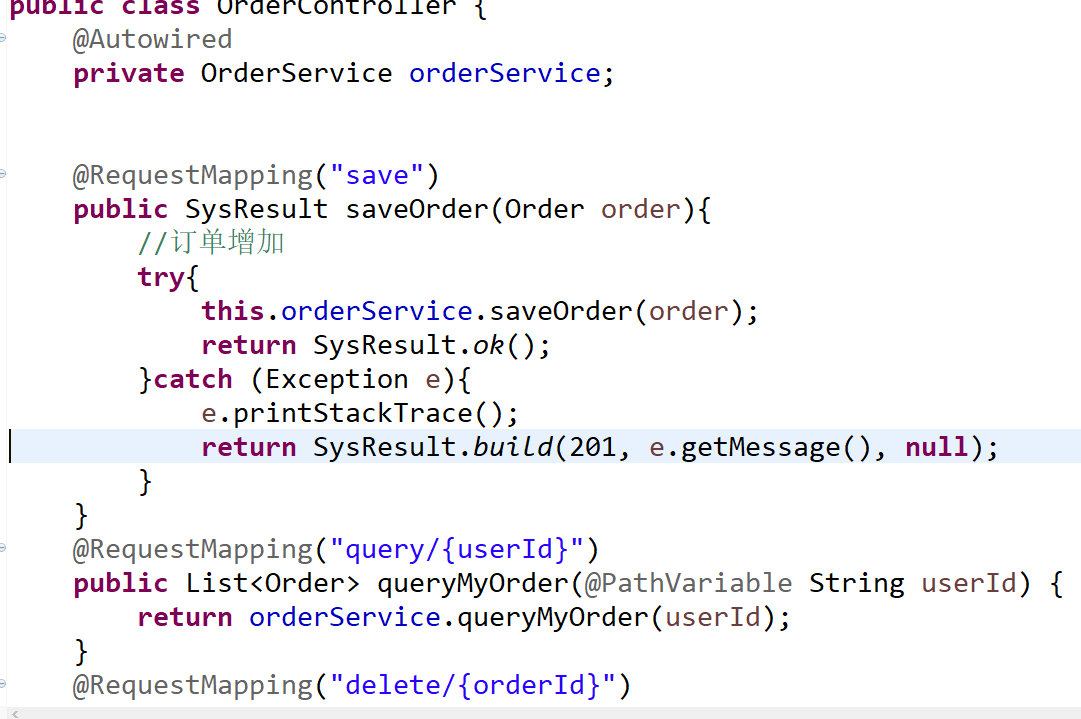


在controller层实现查询，保存，增加，删除购物车商品的业务的调度，之后再service层实现查询库存，核对购物车商品，更新商品数量，删除信息等的一系列具体业务逻辑。

1. 订单系统的实现
2. 配置各种文件

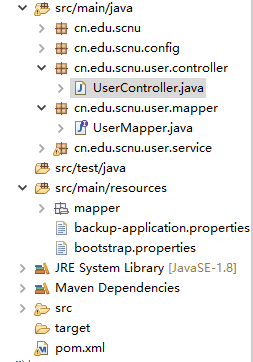


1. 建立dao层，service层，controller层，实现订单展示，添加订单，删除订单的功能。



在controller层实现订单查询，保存，以及订单删除的业务调度，之后再service层实现订单创建，订单信息处理，订单删除，订单落库等的一系列具体业务逻辑。

1. 用户系统的实现
2. 配置各种文件



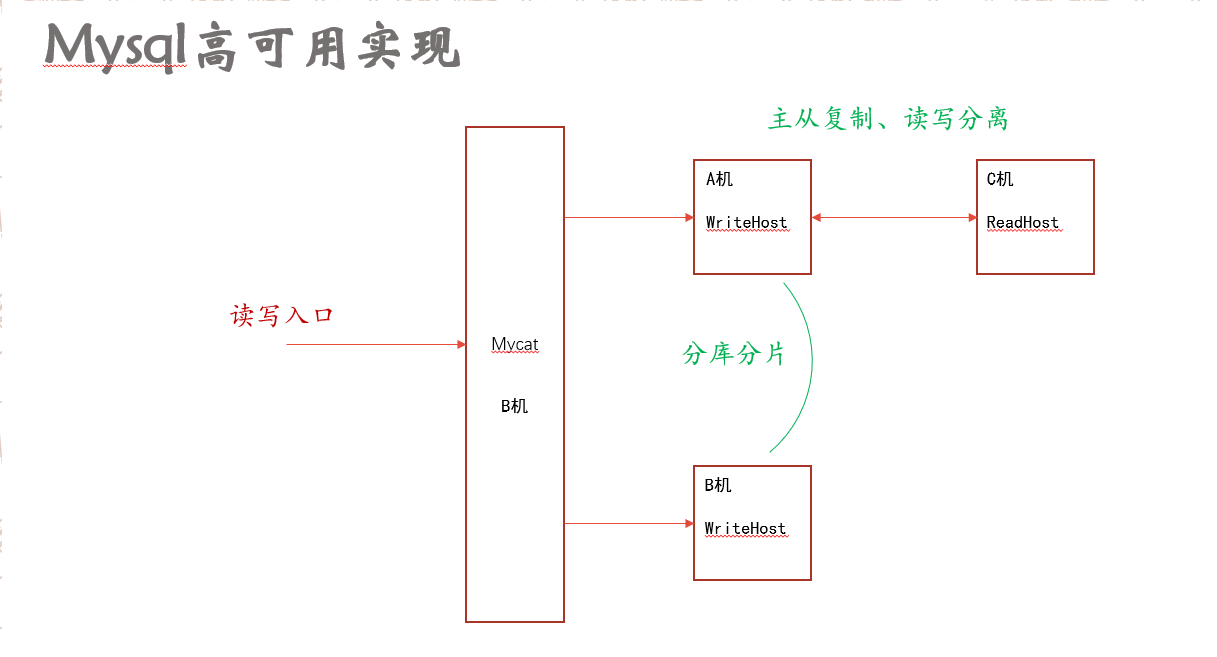
1. 建立dao层，service层，controller层，实现用户登录超时，一个账号最多3个用户同时登录



在controller层实现账号查询，账号注册，账号登录，账号退出等业务的调度，在service层实现账号登录超时，一个账号最多三个用户登录，账号redis缓存获取，redis缓存插入和删除，账号信息核对，账号信息入库等的一系列具体业务逻辑。

### 4.2.2Mysql

Mysql高可用实现总体框架



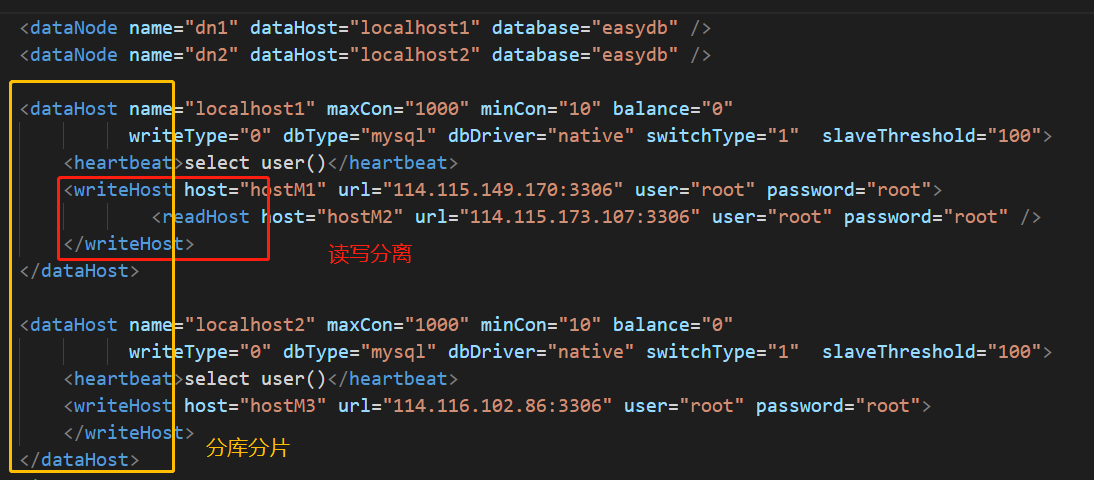
小组共买了3台华为云云上服务器，Mysql的安装和数据存储都是在这个云上服务器里。

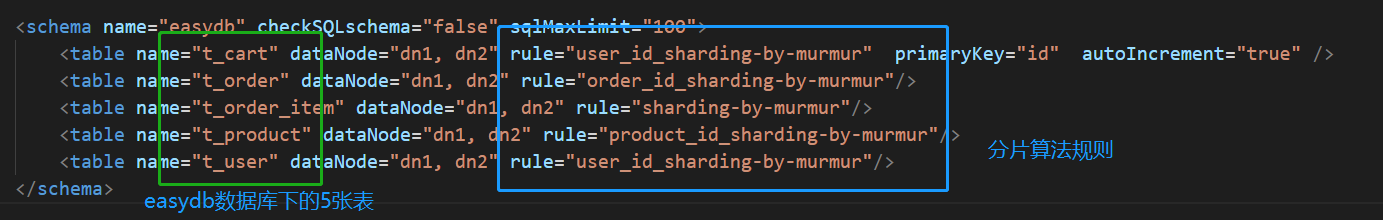
Mycat中间件作为读写入口可以装在任意一台机，不影响同机的Mysql运作，所以这里我安在了B机上。Mycat是数据库中间件，可实现高可用（解决数据量太大，高并发等问题），功能包括：主从自动切换（双机热备）；读写分离（负载均衡）；数据分库分表等。

分库分片：我们将A机和B机进行分库分片。这里全部表是按字符串类型的xxid作为索引分片，采用应用于字符串murmur分片算法。当一个数据插入时，根据索引的值再通过算法决定这条数据应插入在哪架机子中，虽然一张表数据分库存储，但最后通过Mycat查询表时，是将所有分片过的表合并后返回全部数据。

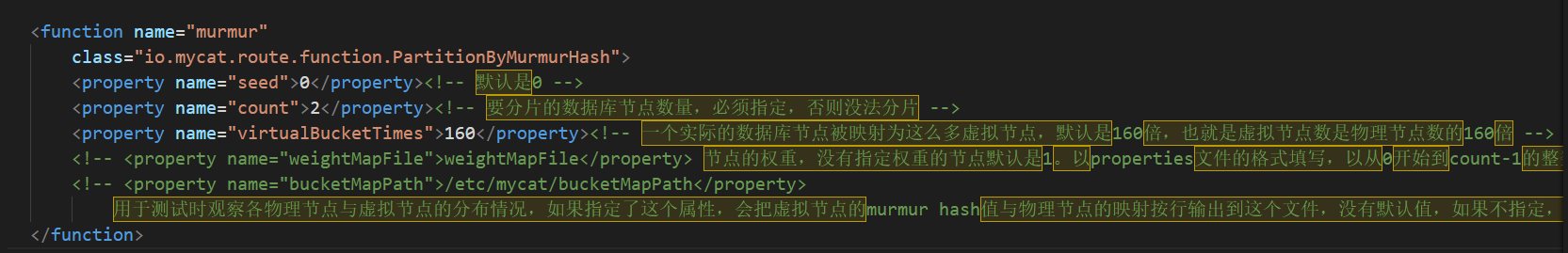
读写分离：我们将A机和C机实现读写分离，一个作为写主机、一个作为读主机。实际大型项目中一个写主机配对多个读主机，因为实际应用中查询次数多于增删次数。读写分离分摊了数据库的压力，做到一个机器上数据库专职某项工作，提高整体效率。

下图Mycat的 schema.xml配置高可用

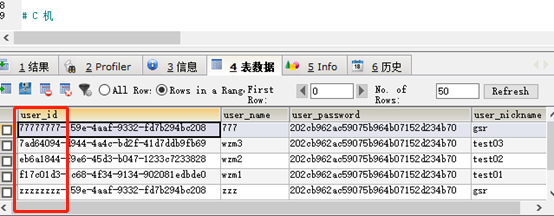


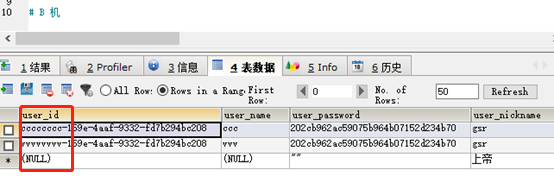


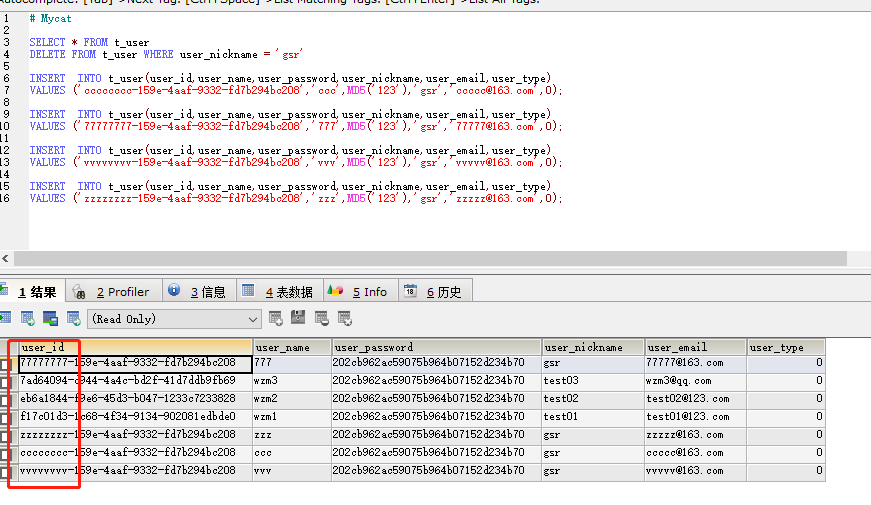
下图rule.xml文件里，采用应用于字符串murmur分片算法



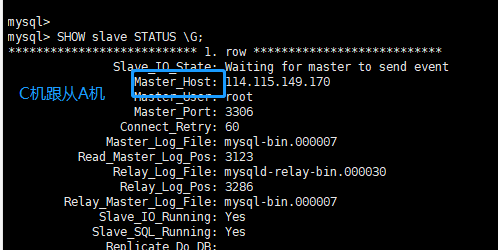
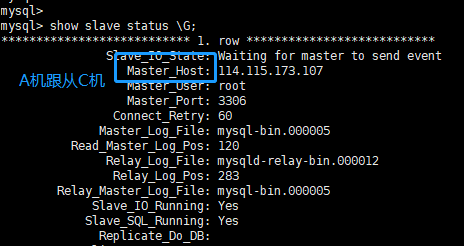
读写分离实现效果







读写分离的基础上要做到主从复制，我们将A机和C机配置成主从复制效果，无论从两架机的哪架机进行增删改查，都能做到数据同步。

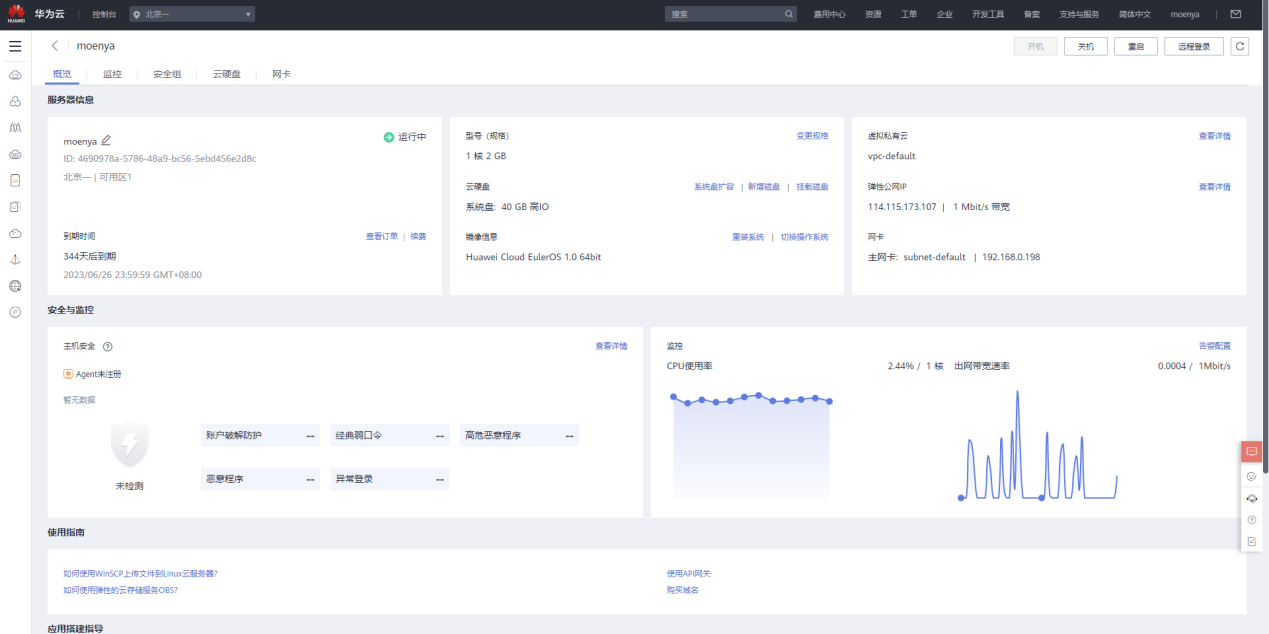


### 4.2.3Redis

1. Redis相关依赖的下载配置

使用华为云服务器，对云服务器进行配置，安装Redis，下载相关依赖。

服务器预览如下：



配置安全组规则，特别是入方向规则，其中包含允许xshell的远程连接SSH的端口号以及相应的网卡端口号等协议和端口。



安装相关依赖略，最终结果如下：



1. 配置多节点Redis

如果一个服务器中只启动一个redis的server。不足以充分发挥一个服务器的性能，所以一般在同一台服务器上,我们可以根据服务器性能多启动几个redis实例,这样一个服务器可以运行多个Redis节点，这时要保证多个节点的端口不能冲突。

Redis运行在保护模式下，保护模式默认是启用的，没有指定绑定地址，没有认证密码要求。在这种模式下，连接只接受环回接口。如果想从外部电脑连接到Redis是不行的，远程链接不能进行 CRUD 等操作。为了保证多个节点端口不冲突，要修改redis配置文件。

1. 注释掉bind，以取消链接限制



1. 关闭保护模式，以取消每次连接时都需要提供登陆密码的限制



1. 修改端口



1. 设置客户端空闲时间，指定为3600秒后才断开



1. daemonize 设置成yes，让redis服务器启动有守护进程管理



1. 设置pid文件名和logfile，每一个端口相对应





1. 配置save策略



1. 指定dump的持久化文件，每个服务单独指向一个文件



1. 将该模板保存，并拷贝为redis01.conf、redis02.conf、redis03.conf 三份，并修改端口号。



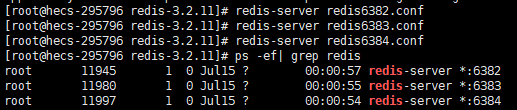
1. 启动三个节点即可，配置完成。



1. 配置Redis的一主二从关系
2. 配置三个端口的配置文件为redis6382.conf、redis6383.conf、redis6384.conf



1. 启动三个节点



1. 默认情况下，每个节点都是master，通过修改配置文件定义主从，并通过命令挂接主节点，显示从节点信息如下：



可以看出已经有两个从节点了。

1. 配置Redis哨兵监听主从

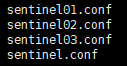
哨兵监听的主从关系是状态。每一次角色的变化都由哨兵来监控——对所有的主从结构的决策都都需要投票。

只有一个哨兵节点的管理结构是不可信，因为如果当哨兵连接主从结构的网络异常时，会误判为主从结构的节点出问题。所以引入哨兵集群。

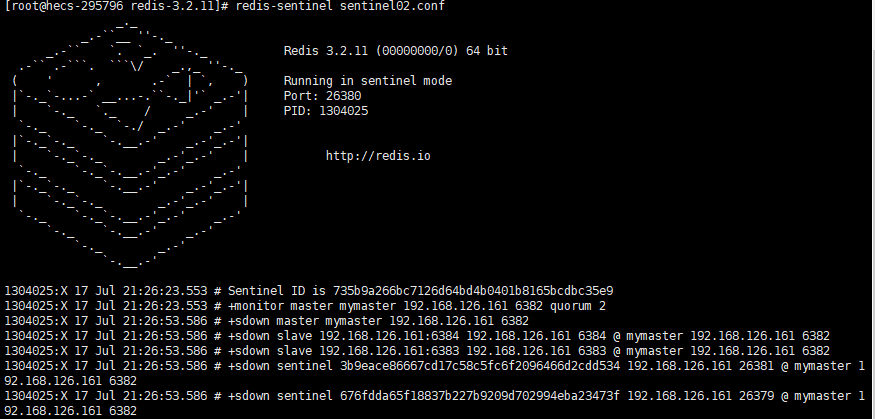
搭建3个哨兵节点,管理6382,6383/6384的主从结构。

1. 配置三个哨兵的端口26379 26380 26381

配置bind的ip地址绑定，设置protected-mode为no，设置端口号，并设置监听主节点的配置核心内容。



1. 先启动主从结构中的三个节点：6382,6382.6384，并启动三个哨兵集群，为了查看日志信息，每个哨兵使用一个Xshell窗口。可以看出每个哨兵相互发现



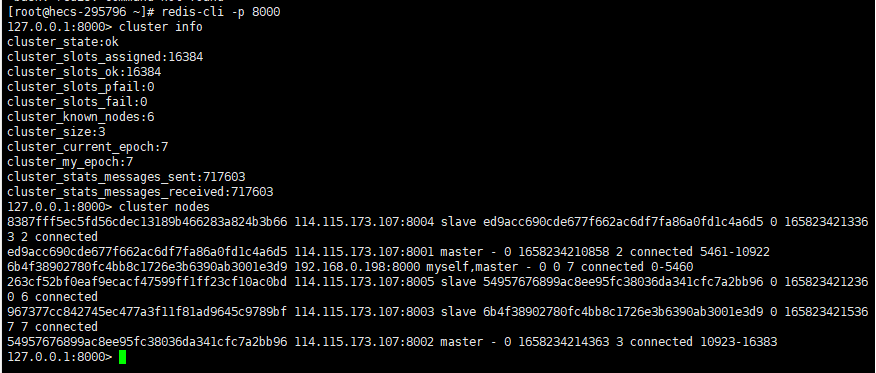
1. 配置Redis集群

Redis3.0有一个一个新的redis集群结构,引入了很多新的概念和逻辑。我们配置该集群以将主从复制、哨兵监听的逻辑整合到了这个该结构中。

1. 配置Ruby，安装Redis接口，详略。
2. 配置集群配置文件，以8000-8005命名。追加模式AOP的持久化策略，开启AOP模式的持久化数据，并指定一个端口号对应的持久化文件；将节点启动定义为集群模式，并打开指定一个集群节点运行状态记录文件conf。



1. 使用ruby脚本命令创建一个3主且各自有一从的集群结构。最终配置结果如下：



1. Redis整合登录

通过点击登录发起ajax请求,将用户名传递给后台处理，根据响应结果判断处理，如果成功跳转向首页。

1. 在用户中心的pom文件添加redis依赖

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-redis</artifactId>

<version>1.4.7.RELEASE</version>

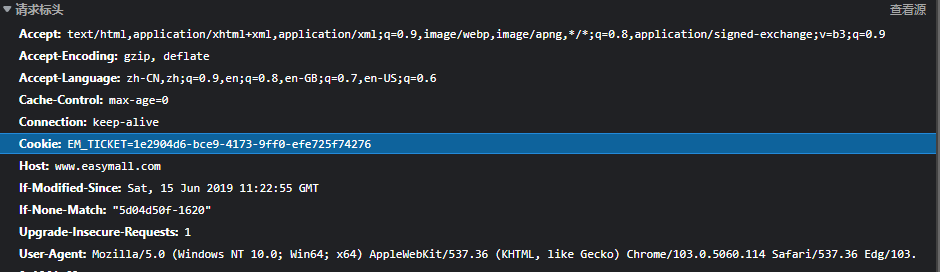
</dependency>

1. 修改UserController、UserService、UserMapper和映射文件UserMapper。
2. 在Zuul网关的application.properties中添加敏感头为空的配置：

zuul.sensitive-headers=

1. 配置properties属性值，编写配置类，编写一个连接池的初始化方法。
2. 修改User的登录逻辑。

可以看出，浏览器存储了密码的Cookie，说明Redis正常工作。



# 5.项目分工说明

## 5.1组员分工

小组共有五名成员，组长为顾思睿，成员有黄海峰、李牧林、王哲孟、何昕宇。

顾思睿：作为组长规划整个项目进程和审核。负责在华为云服务器安装部署Mysql、Mycat，实现主从复制、读写分离、分库分片功能。

黄海峰：负责SpringCloud框架中依赖资源的管理，高可用双注册中心的搭建，高可用zuul网关集群的实现，商品微服务、商品图片微服务的搭建，以及Nginx的搭建与配置。

李牧林：FoodStore生鲜商城前端代码的编写、调试和修改，以及和负责后端的同学沟通如何进行前后端的交互。

王哲孟：负责config配置中心的搭建，公用工程rediscluster，购物车系统，订单系统以及用户微服务的实现。

何昕宇：负责华为云服务器上的Redis主从复制、多节点、哨兵的配置，配置Redis集群进行管理，配置完成后整合登录功能使得密码存储在浏览器cookie中。